Projekt Dreiecksberechnungen

Schreiben Sie ein Programm, welches Flächeninhalt und Umfang eines beliebigen Dreiecks mit einzugebenden Seitenlängen berechnet. Außerdem soll überprüft werden, ob sich mit den vorgegebenen Seitenlängen überhaupt ein Dreieck konstruieren läßt.

---

1. Starten Sie Delphi, wählen Sie **Datei Neue Anwendung** und speichern Sie mit **Projekt speichern unter**. Legen Sie dabei einen **Ordner „Dreieck“** unter Ihrem Namen an.

   Geben Sie dem Anfangsfenster FORM1 die **Caption "Dreiecksberechnungen"** und den **Namen "FoDreieck".**

2. **Erstellen Sie die Form analog zum nebenstehenden Layout**
   (Achtung: Halten Sie sich nicht unnötig lange (max. 15 min) mit der Layout-Erstellung auf.)

   Verwendete Komponenten:

   - Panel für die Überschrift,
   - GroupBox für Eingabe und Ausgabe,
   - Edit Felder für Ein- und Ausgabe (Edit.Text bitte mit sinnvollen numerischen Startwerten belegen)
   - Button für Berechnungsstart

---

3. Deklarieren (*Private declarations*) und implementieren Sie die Funktion

   ```
   function TFoDreieck.Umfang(a,b,c: Real):Real;
   ```

   welche aus den Übergabewerten a, b und c (Seitenlängen des Dreiecks) den Umfang des Dreiecks a+b+c als **Rückgabewert Result** liefert.

---

4. Testen Sie Ihre selbstdefinierte Funktion, indem Sie im "Hauptprogramm"

   ```
   procedure TFoDreieck.BuBerechneClick(Sender: TObject);
   ```

   - die Text-Information der Edit-Eingabefelder Seite a, Seite b, Seite c mittels der Standardkonvertierungsfunktion

     ```
     StrToFloat(s: String): real;
     ```

     in Fließkommazahlen a, b, c umwandeln und

   - das Ergebnis des Funktionsaufrufs Umfang(a,b,c) mit der inversen Konvertierungsfunktion

     ```
     FloatToStr(x:real): String
     ```

     als Text an das Ausgabefeld Umfang übergeben.

5. Deklarieren und implementieren Sie die Funktion

```
function TFoDreieck.Flaeche(a,b,c: Real):Real;
```

welche aus den Übergabewerten a, b und c (Seitenlängen des Dreiecks) den Flächeninhalt des Dreiecks als **Rückgabewert Result** liefert.
Zur Berechnung der Dreiecksfläche benutzen Sie die **Heron'sche Dreiecksformel:**

$$A = \sqrt{s \cdot (s-a) \cdot (s-b) \cdot (s-c)} \qquad \text{mit} \qquad s = \frac{a+b+c}{2}$$

die ganz ohne Winkelfunktionen auskommt. s ist dabei die Hälfte des Umfangs.
Benutzen Sie die mathematische Standardfunktion **`Sqrt(x:Real):Real`**
und die eben definierte Umfangsfunktion **`Umfang(a,b,c: Real):Real`**.

---

6. Testen Sie diese Funktion analog zur Teilaufgabe 4.
Für eine formatierte Ausgabe von Umfang und Fläche mit zwei Nachkommastellen verwenden Sie die Standardfunktion

```
FloatToStrF(Umfang (a,b,c),ffFixed,10,2);
FloatToStrF(Flaeche(a,b,c),ffFixed,10,2);
```

Konsultieren Sie die Delphi-Hilfe, um die Bedeutung der einzelnen Parameter zu erfahren und benutzen Sie dann das fixed point format mit 10 Ziffern und 2 Dezimalstellen. Testen Sie mit unterschiedlich großen Seitenlängen. Sollte die Fehlermeldung NAN erscheinen, konsultieren Sie die Delphi-Hilfe.

---

7. Deklarieren und implementieren Sie die Funktion

```
function DreieckProbe(a,b,c: Real):Boolean;
```

welche überprüft, ob mit den eingegebenen Seitenlängen a, b, c ein Dreieck konstruierbar ist. Dabei müssen je zwei Seiten zusammen stets größer als die dritte Seite sein (Dreiecksungleichung). Ist diese Bedingung erfüllt bzw. nicht erfüllt, so nimmt der **Rückgabewert Result** den Wert True bzw. False an. True und False nennt man Wahrheitswerte (**Datentyp Boolean**).

---

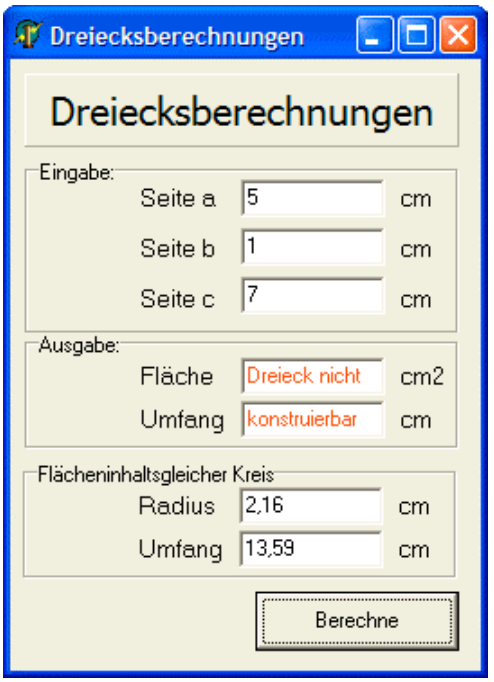
8. Geben Sie für den Fall, daß das Dreieck nicht konstruierbar sein sollte, eine entsprechende **Fehlermeldung in roter Schrift** (Eigenschaft Edit.Font.Color) in den Ausgabefeldern Umfang und Fläche aus (siehe Screenshot).

---

9. Ergänzen Sie Ihre Form durch eine zusätzliche **GroupBox "Flächeninhaltsgleicher Kreis",** in der Sie die Rückgabewerte der beiden neuen Funktionen

```
function KRadius (f: Real):Real;
function KUmfang(f: Real):Real;
```

ausgeben (siehe Screenshot).



Standardfunktionen in Delphi

Mathematische Funktionen:

Abs(x:real) :real
Sqrt(x:real):real
Exp(x:real) :real
Ln(x:real):real
Cos(x:real):real
Sin(x:real):real
Tan(x:real):real
ArcTan(x:real):real

Funktionen zum Typ String:

Copy (s:string a,b:integer): string gibt Teilstring eines Strings zurück.
Bsp: copy('abcde',2,3) = 'bcd'

Length(s:string):integer
gibt die Anzahl der Zeichen eines String zurück.
Beispiel: length('abcde') = 5

Pos (a,b:string) :integer
gibt den Indexwert des ersten Zeichens von a zurück, der in dem String b vorkommt (sonst =0)
Beispiel: pos('ef','aebcdef') = 6

Chr(n:integer):char
gibt das Zeichen eines ASCII-Wert zurück.
(0<=n<=255). Umkehrfunktion ist ord(n).
Beispiel: chr('a') = 97 chr('A') = 65

Ord(c:char): integer
Die Umkehrfunktion von chr(n)
Beispiel: ord(97) = 'a' ord(65) ='A'

Funktionen zum Typ Real:

Frac(x:real): real
gibt den Nachkommateil einer Zahl zurück. Frac(2.4578) = 0.458

Int(x:real): real
gibt den ganzzahligen Anteil einer Zahl zurück. Int(4.899) = 4

Round(x:real) :integer
rundet den Wert von x
auf den nächsten Integer-Wert.
Round(4.5351) = 5
Round(4.4535) = 4

Trunc(x:real): Integer
konvertiert eine Gleitkommazahl in einen Integer-Wert.
Beispiel: Trunc(4.983) = 4

Konvertierungsfunktionen

StrToInt(s: String):integer
StrToInt konvertiert einen String, der einen Integer repräsentiert, in eine Zahl

StrToFloat(s: String): real
StrToFloat konvertiert einen String in einen Gleitkommawert.

IntToStr(n:integer): String
IntToStr konvertiert den Integer-Wert n in einen String.

FloatToStr(x:real): String
FloatToStr konvertiert Gleitkommazahl x
in die entsprechende String-Darstellung.

Weitere Funktionen

formatfloat('### ### ##0.### ### ###', x: real) :string ()
konvertiert die Gleitkommazahl x in die entsprechende String-Darstellung.

Random(n:integer):integer
erzeugt die Zufallszahl 0 oder höchstens n-1
Beispiel: Wuerfelaugenzahl :=Random(6) + 1

Randomize
initialisiert den Zufallszahlengenerator mit einem zufälligen Wert